Simple Java-RMI Example

First of all, let's create an interaction diagram that demonstrates how the Client and RObject communicate with each other through RegisterService:

sequenceDiagram participant C as Client participant RS as RegisterService participant RO as RObject participant RMI as RMI Registry RS->>RMI: Register RObject RMI-->>RS: RObject Registered C->>RMI: Lookup RObjectServer RMI-->>C: Return RObject C->>RO: primitiveArg(2012) RO-->>C: Acknowledge C->>RO: argumentByValue(2012) RO-->>C: Acknowledge

This diagram illustrates the steps of the process: RegisterService registers the RObject in the RMI Registry, the Client queries the remote object, then uses its functions.

Now let's see what this system looks like in a structure diagram, which shows the components and their relationships:

graph TD subgraph "RMI Registry" RO[RObject Interface] end subgraph "Server Side" ROS[RObjectImpl] -- Implements --> RO end subgraph "Client Side" C[Client] -- Uses --> RO end RegisterService -- Registers --> RO C -- Looks up --> RO C -- Calls methods on --> ROS

The structure diagram clearly depicts the different parts of the system and the relationships between them:

The Server Side section contains the RObjectImpl implementation, which implements the RObject interface. The RMI Registry registers the interface, allowing the Client Side Client to find and use it.

The Client calls the methods of the remote object (ROS), illustrating the two main operations: passing a primitive and passing by value.

Implementation:

Install Gradle from here: https://gradle.org/releases/ Add the *bin* directory to the *PATH*.

The complete source code is available here:

```
git clone https://github.com/knehez/isi.git
cd java_rmi
```

Open two terminals, in one of them:

```
gradle runRegisterService
```

In the other one:

```
gradle run
```

The program in detail:

1.) Define the **RObject** interfaces, both implement the Remote interface

```
package org.ait;
import java.rmi.*;

public interface RObject extends Remote {
  // simple argument passing
  void primitiveArg(int num) throws RemoteException;
  // pass by value argument
  void argumentByValue(Integer num) throws RemoteException;
}
```

2.) Implement the code for remote objects

```
package org.ait;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class RObjectImpl extends UnicastRemoteObject implements RObject {
      private static final long serialVersionUID = 6350331764929058681L;
      public RObjectImpl() throws RemoteException {
      }
      @Override
      public void primitiveArg(int num) throws RemoteException {
         System.out.println(num);
      }

      @Override
      public void argumentByValue(Integer num) throws RemoteException {
         System.out.println(num);
      }
}
```

3.) Start the RMI registry. This is a **Java JDK** component, to start it you should be in the created project's **/bin** directory.

4.) Create an instance of the remote object and bind it to the registry

```
package org.ait;

import java.rmi.Naming;

public class RegisterService {
    /**
     * @param args
     */
    public static void main(String[] args) {
```

```
        try {
            RObject robj = new RObjectImpl();
            Naming.rebind("rmi://localhost:1099/RObjectServer", robj);
            System.out.println("Registered...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Start the RegisterService, it can only register if the rmiregistry is running.

5.) Use the remote object

Start the following code.

```
package org.ait;

import java.rmi.Naming;

public class Client {
    public static void main(String[] args) {
        try {
            // Retrieving the remote object from the registry
            RObject robj = (RObject)
Naming.lookup("rmi://localhost:1099/RObjectServer");

            // Simple argument
            robj.primitiveArg(2012);

            // Serialized argument
            robj.argumentByValue(new Integer(2012));

        } catch (Exception e) {
            e.printStackTrace();
            ;
        }
    }
}
```